Keyword Detection in Text Summarization

Justine Raju Thomas
710CS1118
NIT Rourkela

A thesis submitted for the degree of
Dual Degree in Computer Science

May 2015

Every challenging task requires two things : Effort from self and the support and guidance of ones elders.

Saying so, I dedicate this thesis to my parents whose love, sacrifice and prayers have made me able to achieve such success and honour.

Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India. www.nitrkl.ac.in

Dr. Korra Sathyababu
Professor

May 23, 2015

Certificate

This is to affirm that the work presented in the thesis entitled *Keyword Extraction in Text Summarization* by **Justine Raju Thomas** is a record of genuine research work carried out by him, under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of Dual Degree(B.Tech + M.Tech) with the specialization of Computer Science in the department of Computer Science and Engineering, National Institute of Technology, Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: NIT Rourkela
Date: 23 May 2015

Korra Sathyababu
CSE Department
NIT Rourkela

Declaration

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions. I hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any award. The interpretations put forth are based on my reading and understanding of the original texts and they are not published anywhere in the form of books, monographs or articles. The other books, articles and websites, which I have made use of are acknowledged at the respective place in the text. For the present thesis, which I am submitting to NIT Rourkela, no degree or diploma or distinction has been conferred on me before, either in this or in any other University. I bear all responsibility and prosecution for any of the unfair means adopted by me in submitting this thesis.

Signature :

Date :

Acknowledgements

I have had the good luck of having many people supporting me during the entirety of my research.

I would like to begin by thanking my guide Prof. K Sathyababu who not only guided me in my works, but also was patient and understanding when things weren't going according to plan.

I would also like to thank my friends for their wholehearted support and friendship. Abhishek, Dev and Subash deserve a special mention for their help during the project.

I would also like to thank Santosh Bharati for proof reading my thesis and correcting me whenever needed.

And lastly but not the least, I thank my parents, for their constant encouragement, love and support and for standing by me always.

Abstract

Summarization is the process of reducing a text document in order to create a summary that retains the most important points of the original document. As the problem of information overload has grown, and as the quantity of data has increased, so has interest in automatic summarization.

Extractive summary works on the given text to extract sentences that best convey the message hidden in the text. Most extractive summarization techniques revolve around the concept of finding keywords and extracting sentences that have more keywords than the rest. Keyword extraction usually is done by extracting important words having a higher frequency than others, with stress on important. However the current techniques to handle this importance include a stop list which might include words that are critically important to the text.

In this thesis, I present a work in progress to define an algorithm to extract truly significant keywords which might have lost its significance if subjected to the current keyword extraction algorithms.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

"Text summarization is the process of distilling the most important information from a text to produce an abridged version for a particular task and user "

-Mani and Maybury, 1999 [1]

Information overload in today's time has called into attention a need for a tool that extracts out only needed information from the documents at hand. As a result for the search for such a tool, came the concept of Text Mining. Text mining, as the name itself suggests, is the process of mining large quantities of text to derive high quality information. These high quality information is usually derived through pattern recognition approaches and statistical information. Text mining includes tasks such as text summarization, sarcasm detection, keyword extraction etc. We concentrate upon two major tasks handled by Text mining : Keyword Detection and Text Summarization.

1.1 Motivation

In days where power means everything, Information is the most powerful tool one can have. And information is available in unfathomable depths each day in the form of electronic newspaper articles, emails, web pages and search results. However information we receive is seldom complete, such that further cleaning activities are required to facilitate correct interpretation and usability of this information. For instance, in any enterprise, given a request sent via e-mail, cleaning activities in the customer support department can include lookups of the information on similar products in the intranet databases, involving Web- and desktop search as well.

By enabling efficient scanning of large document collections, keyword search has become a very powerful tool. The user is freed from the troubles of learning the syntax of a structured query language, and understanding their complex semantics. Amongst recent development, keyword search has been valued immensely in database development, where it enables data retrieval in cases where the schema is hidden from or is obscure to the user, thus going beyond conventional applications. However, usability comes at a cost; the cost of expressiveness. Identifying the users intentions behind supplying the keywords has to be understood in order to correctly answer a

keyword request; thus introducing additional query processing cost at the server side.

Now think of a case where the information needed by a user is represented through a document, instead of a manually created keyword list. Herein lies a problem for keyword extraction, which can be extracted using keyword annotations of this document (along with other available metadata). For example, research articles and papers are often annotated with keywords. Web documents can also be already associated with tags. In electronic media, keywords help a user determine the main idea of an article. In e-books, they help user retrieve the information sought faster. On the web, keywords help find resources and multimedia using annotated tags. However, a huge portion of articles on the web havent been tagged. And manually annotating every article is, if not impossible, very difficult; requiring the needs of automated keyword extraction. To further stress upon the needs of an automated keyword extraction scheme, consider the following scenario.

A technician Bob works for an internet hardware sales company that supplies customers with expert guidance regarding installation and usage of their products. Every day his email account is flooded with mails containing description of the products and usage problems at various levels of detail. To process these requests, he has to extract product details like model no, manufacturer, production country, etc. and then search accordingly in a database for further details. To service these requests, hed have to read the message, trying to identify the useful information or keywords, and then retrieve all the necessary information from the database using a keyword search algorithm. However manually assigning highly valued keywords is a very time taking and tedious task. Automatic keyword extraction would empower Bob to identify keywords from the requests and immediately retrieve information from the server. Many algorithms have been proposed for the extraction of keywords. [1, 2, 3, 4, 5, 6, 7, 8, 9] Currently existing algorithms require domain specific knowledge. For example : It needs to know what the article is about. And most of them have failed to produce a good result and accuracy and hence, are generally not considered trustworthy.

In this thesis, we develop an idea to identify keywords from text documents. We analyzed document statistics, which are useful for keyword generation and keyword disambiguation within an article. We have also compared the proposed idea against the existing algorithms against news articles.

1.2 Outline

The thesis is organized as follows

Chapter 1 dealt with an introduction to the topic presenting the motivation for doing this project.

Chapter 2 analyzes the problem and touches on the existing algorithms in the field of keyword extraction and text summarization. This slowly leads us to our conceptual design.

Chapter 3 deals with existing methodologies that have been made use of during the implementation of this project.

Chapter 4 exhibits the intrinsic details of my proposed work. The work includes a proposed algorithm for keyword extraction in section 4.1 and an algorithm for

summarization in section 4.2

Chapter 5 describes the structure of the data-sets involved.

Chapter 6 takes up a case study and explains each step of the proposed algorithm step by step for proper understanding.

Chapter 7 evaluates the proposed algorithm and compares the results against the existing algorithms.

Chapter 8 gives a conclusion to the work done and supplies a direction for the future work.

Chapter 9 gives a list of the references.

Chapter 2

Problem Analysis

The focus of our work lies in enabling a user to search for keywords from within a text file and help him summarize the entire document using those keywords. To facilitate this, we have broken the process into two major parts: firstly the extraction of keywords in which the information of the document is concentrated, and then effectively summarize the document. Therefore we have divided this chapter into two parts: The section 2.1 gives a brief introduction to keyword extraction whereas section 2.2 deals with the problem of summarization. Each section also touches upon existing algorithm to help the user familiarize with related works and the logic behind our conceptual design

2.1 Keyword Extraction

Automatic keyword extraction is the process of selection small words and phrases from a document that can best project the core sentiment of the document. And as the name suggests, the extraction is done automatically without any human intervention depending on the model. The target of automatic keyword is the application of the power and speed of modern computation abilities to the problem of access and recovery, stressing upon information organization without the added costs of human annotators.

2.1.1 Existing Approaches

Manual Extraction or annotation of keywords is a tedious process brimming with errors involving lots of manual effort and time. Therefore many automatic keyword extraction procedures have been suggested. The algorithms can mainly be categorized into the following categories : Simple Statistics Approach, Machine Learning approach and hybrid approaches.

2.1.1.1 Simple Statistics Approach

These methods are crude, simplistic and tend to have no training sets. They focus on statistics derived from non-linguistic features of the document text such as position of a word within the document, the term frequency and inverse document frequency.

These statistics are later used to develop a list of keywords. Cohen, one of the pioneers in the field, used N-Gram statistical information to automatically find the keywords within a document. Other methods within this category involve TF or the word frequency, TF*IDF. These algorithms are faster and tend to give a good result.

Most of the approaches lie in this category. The most basic of them is TF. In this algorithm, the frequency of occurence is the only criteria that decides whether a word is keyword or not. This algorithm is very crude and tends to give quite horrible results.

An improvement to this algorithm is the TF-IDF which too takes in frequency of occurrence of a word as the criterion to decide whether a word is keyword or not. However, as an added feature, we also take into consideration the rarity of the word in other texts as a criterion too. That is a word that is highly common in the article under consideration and is hardly found in other documents gets very high score so that only the words pertinent to this article gets chosen as keywords.

2.1.1.2 Pattern Recognition Approaches

Keyword extraction can also be seen as a learning problem. Use of manually annotated training data and training models like Hidden Markov Model, SVM, Nave Bayes etc. are commonly used in these approaches. In the second phase, the document whose keywords need to be extracted is inputted to the model, which then extracts the keywords which best fit the models training.

One of the most famous algorithms in this approach is the keygraph algorithm. The article is first converted into a graph. Each word is treated as a Node and whenever two words appear in the same sentence, the nodes are connected with an edge for each time they appear together. Then the number of edges connecting the vertexes are converted into scores and is clustered accordingly. The cluster heads so achieved are treated as keywords.

Similarly, word co-occurrences algorithm, deal with statistical information of the number of times a word has occurred and the number of times it has occurred with another word. This statistical information is then used to calculate support and confidence of the words. Apriori method is then used to derive the keywords.

Bayesian algorithms uses the bayes classifier to classify the word into two categories -keyword or not keyword depending on how it is trained. So are other classifiers in this approach.

2.1.1.3 Hybrid Approaches

These approaches generally combine the above two methods or use some knowledge or heuristic, such as position, tags etc. These algorithms are generally designed to take the best features out of both the above approaches. Our proposed algorithm follows a mixed approach. We apply natural language processing to identify parts of speech of the terms within an article and then use statistical approach to extract keywords. Morover in our work, we further use these statistics as a means to summarize the text in hand.

2.2 Summarization

Summarization is a process where the most salient features of a text are extracted and compiled into a short abstract of the original text. Summaries are usually around 17% of the original text and yet contains everything that could have been learnt by reading the original article. Thus summarization is a very effective and powerful tool and hence the need for automatic summarization has gone up in a very large scale in todays times where data is too large to go through.

Automatic Summarization basically is of two types : Abstract based Summarizers and Extract Based Summarizers. Abstract Based summarizers is a topic under huge research, however, no standard algorithm has been achieved yet. Abstract based summaries are derived by learning what was expressed in the article and then converting it into a form expressed by the computer. It resembles how a human would summarize an article after reading it.

Extract based summaries [10, 11, 12, 13, 14, 15] on the other hand extract details from the original article itself and presents it to the reader. My work focuses on extract based summaries too. I shall explain a couple of existing algorithms below and later show in my proposed work how my algorithm changes how they work.

First Come First Serve Model :

The first step involves the selection of keywords. Once the keywords are chosen, the algorithm parses through the document to select the sentences in which the keywords occur. Once sufficient number of sentences have been derived, the algorithm exits. This algorithm is based on the fact that the first paragraph of an article usually contains a summary of what is expressed in the document.

TextRank :

As usual, the first step involves selection of the keywords. The algorithm then parses through the document and makes a note of which sentences contain how many keywords. Considering this as a score, the algorithm then sorts them in descending order and extracts the desired number of sentences from the top.

This was a brief introduction to summarization techniques. We conclude this chapter here. The next chapter shall deal with the existing methodologies that we have used in our proposed work.

Chapter 3

Methodologies

Our work makes extensive use of existing methodologies especially those of POS Tagging and Anaphora and Cataphora Resolution. This chapter provides a brief introduction to these topics. Section 3.1 shall deal with POS Tagging and Section 3.2 shall deal with Anaphora and Cataphora resolution.

3.1 POS Tagging

POS Tagging is an area of study that comes under the vast topic of Natural language Processing. So before we step into the field of POS tagging, we'll present a brief summary of the Natural Language Scenarios that lead to POS tagging

3.1.1 The language modelling problem

Let us assume that we have a finite set V, which will include all of the words in our language of interest. Hence,

V=the,a,man, telescope, Beckham, two, ...

Now given the set V, we derive a set V+ which is a set of all possible sentences or strings in this language. V+ may include

the	STOP			
a	STOP			
the	fan	STOP		
the	fan	saw	Beckham	STOP
the	fan	saw	saw	STOP
the	the	the	STOP	
etc				

where STOP is a symbol that is used for our convenience in computations.

Given a training set of example sentences in English, we are required to learn a probability distribution p, such that

$$\sum_{x \in V+} p(x) = 1, p(x) \geq 0 forall x \in V+$$

3.1.2 Parts-of-Speech Tagging

POS tagging is a process that takes as input a sequence of words and gives as output the associated part-of-speech tag for each word.

Example :

Input : Profits soared at Boeing Co., easily topping forecasts on Wall Street.

Output : Profits—N soared—V at—P Boeing—N Co.—N ,—, easily—ADV topping—V forecasts—N on—P Wall—N Street—N

3.1.2.1 Hidden Markov Models

We have an input sentence x= x_1,x_2.....x_n (x_i is the ith word in the sentence) We have a tag sequence y=y_1,y_2....y_n (y_i is the ith tag in the sentence)

We'll use an HMM to define P(x_1,x_2,....x_n,y_1,y_2....y_n) for any sentence x_1,....x_n and tag sequence y_1...y_n of the same length. Then the most likely tag sequence for x is

arg max$_y$ p(x$_1$,x_2....x_n,y_1....y_n)

3.1.2.2 Trigram HMM Models

For any sentence x1...xn where xi $\in$ V for i=1...n, and any tag sequence y_1...$y_n + 1$ where $y_i \in$ S for i=1...n and $y_n + 1$= STOP, the joint probability of the sentence and tag sequence is

p(x_1,..x_n,y_1...y_{n+1}) = $\prod_{i=1}^{n} q$(y$_i$—y_{i-2},y_{i-1}) $\prod_{i=1}^{n} e$(x$_i$—y_i)

Example :

Sentence : the dog laughs
Tag : D N V STOP

P(x_1,x_2,....x_n,y_1,y_2....y_n)= q(D$\|*, *$)$*q(N\|*, D)*q(V\|D, N)*e(the\|D)*e(dog\|N)*$ $e(laughs\|V)$

3.1.2.3 What are q() and e()

q() is a smoothed estimation as shown below

q(Vt$\|Dt, JJ$) = $\lambda_1 * \frac{Count(Dt,JJ,Vt)}{Count(Dt,JJ)} + \lambda_2 * \frac{Count(JJ,Vt)}{Count(JJ)} + \lambda_3 * \frac{Count(Vt)}{Count()}$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1 \, and \, for \, all \, i, \lambda_i \geq 0$

And

e(base$\|Vt$) = $\frac{Count(Vt,base)}{Count(Vt)}$

3.1.3 The Viterbi Algorithm

Problem : For an input x1....xn , find
argmax $_{y_1....y_k}$ p(x_1,x_2....x_n,y_1....y_{n+1})

Solution :

Define n to be the length of the sequence.

Define S_k for k=-1....N to be the set of all possible tags at position k

So $S_{-1} = S_0 = \{ * \}$
S_k = S for k $\in \{ 1 ... N \}$

Define r(y_{-1},y_0,y_1... y_k)=$\prod_{i=1}^{k} q(y_i \| y_{i-2}, y_{i-1})$ * $\prod_{i=1}^{k} e(x_i \| y_i)$

Let π (k,u,v) = maximum probability of a tag sequence ending in tags u,v at position k.

that is

π (k,u,v) = max $_{y_{-1},y_0,y_1....y_k}$ r(y_{-1},y_0,y_1....y_k)

A recursive solution :
For the base case :

π (0,*,*) = 1;

For any k $\in$ 1...n , for any u $\in S_{k-1}$ and v $\in S_k$

π (k,u,v) = max$_{w \in S_k}$ (π (k-1,w,u) * q(v$\|w, u$) * $e(x_k \| v)$)

3.2 Anaphora and Cataphora Resolution

In linguistics, a part of the text may refer to a part previously mentioned. For example : John was sad that he lost the match; The proper noun *John* and the pronoun *he* refer to the same entity. This is the phenomenon of Anaphora. If the referent is mentioned later, the phenomenon is known as Cataphora. For example : Because he was ill, Sam had to miss the match; *Sam* is the referent and *he* which refers to Sam comes before the referent is used. The process of resolving the Anaphor to the Antecedent is known as Anaphora and Cataphora Resolution.

Anaphora resolution is quite tricky. A human mind can quite easily resolve the anaphors to their right antecedents. However training a computer to do so is quite a task. Consider the following clarifying example from a British World War II anti-raid leaflet: "If an incendiary bomb drops next to you, dont loose your head. Put it in a bucket and cover it with sand." Confused? Indeed "it" could stand for (or refer to) either of the two objects mentioned before it, "bomb" and "head". The authors meant the former, but the rules of language have a tendency to bias readers to picking the latter. What helps us humans resolve the antecedent is the fact that no one ever stuffs their head into buckets and none of us would want to buried alive. This is what is expected out of an automatic anaphora resolution system; The knowledge to add the information already known to the information given and extract something meaningful.

As an example of how tough this can really get, I would like to give another example. Consider the following sentence : "The children had sweets. They were deli_____ ". The last word can be substituted by the words: "delighted" and "delicious". Each word yields a different antecedent for the pronoun "they": "the children" and "the sweets" respectively. The researches that have worked on this direction can be referred to from the Bibliography. [16, 17, 18, 19, 20, 21]

This concludes an introduction to the existing methodologies that we have heavily used for the progress of our work. The next chapter shall explain in detail how our proposed scheme works and how the methodologies explored in this chapter were used in the project.

Chapter 4

Proposed Work

This chapter deals in explicit details on the approach we used for keyword extraction in section 3.1 and summarization in section 3.2

4.1 Automatic Keyword Extraction

The aim of automated keyword extraction is to point out a set of words or phrases that best represents the document. To do so, we have proposed a hybrid extraction technique. This is done in steps which have been described below.

4.1.1 Training

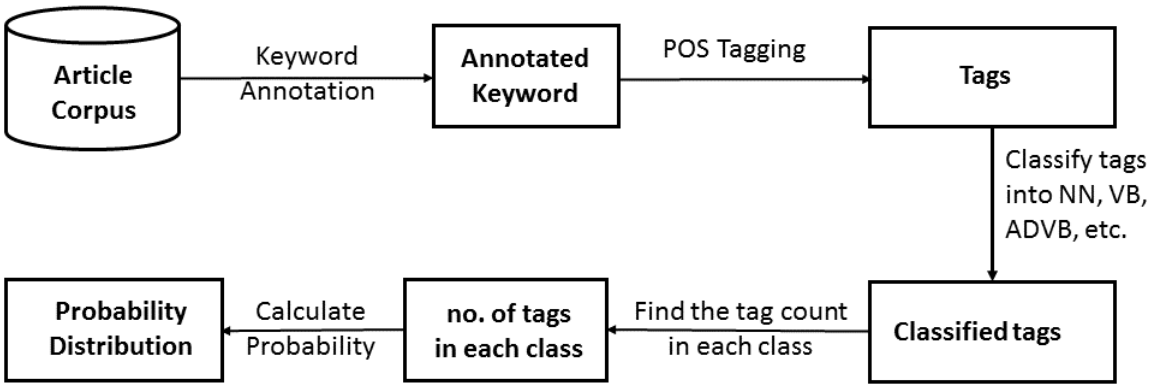


Figure 4.1: Model for Training

Given above is the model for training the model we are going to use. We shall explain the important parts of the model in the following subsections.

4.1.1.1 Parts-of-speech Tagging

English Grammar deals with many parts of speech like nouns, verbs, adjectives, prepositions etc. The automatic labelling of words to their parts of speech is known as POS

Tagging.

POS tagging is a supervised learning problem. We have used Hidden Markov Model to analyze the POS tags. We have followed the brown corpus style of tagging having 44 tags. Hidden Markov Models count the number of cases that arises and creates a table of probabilities from it. As an example : Once you encounter a determiner, such as the, maybe the probability that the next word is a noun is 40% and it being a verb is 20%. Once the model finishes it training, it can be correctly used to determine whether 'can' in 'the can' is a noun (as it should be) or a verb. We have worked on trigram HMMs and the accuracy came out to be around 75-80%.

For our model, we require human intervention in order to train our algorithm. The human annotators analyze documents and select probable keywords. These keywords are supplied to the POS tagger with respect to the documents and the output is supplied to the next section of the model.

4.1.1.2 Learning Probability Distribution

Due to lack of reliable human annotators, we used newspaper clippings as our training dataset. The article was considered as the target document and the headlines as the keywords, thus eliminating the need of human annotators. Now we study this training dataset and find out the number of Nouns, Verbs, Articles, Adjectives etc appeared as a keyword in the headlines. After studying a dataset of 600 articles - a total of 2678 keywords, here are the results we got.

VB	222
NN	926
VBD	206
NNP	342
NNS	343
JJ	328
VBG	144
JJS	76

Table 4.1: Number of times a tag has been found in Newspaper Headlines

We now use these studies as a probabilistic measure to detect keywords. To convert them into probabilistic values, we divide the counts by 2678 which gives us

VB	0.082
NN	0.363
VBD	0.077
NNP	0.1277
NNS	0.1441
JJ	0.1224
VBG	0.053
JJS	0.028

Table 4.2: Probability scores derived from table 3.1

This gives us a trained probabilitistic measure. Lets call it P(tag).

Algorithm 1: Algorithm to train probability distribution

Data: *dataset* := dataset of articles
keyword := Human annotated set of keywords for each article.
Result: P(*tag*) : Probability Distribution of Tags

```
1  count=0;
2  while tag in taglist do
3  |  Tag_count(tag)=0;
4  end
5  while article in dataset do
6  |  while keyword in article do
7  |  |  tag=POS_tag(keyword);
8  |  |  Tag_count(tag)=Tag_count(tag) + 1;
9  |  |  count=count+1;
10 |  end
11 end
12 while tag in taglist do
13 |  P(tag)=Tag_count(tag)/count;
14 end
```

Given above is an algorithm that derives P(tag) from the dataset. Input provided to the algorithm are a dataset of articles along with human annotated keywords for each article.

Line 1 initializes the variable *count* to 0. This variable stores the number of keywords that has been scanned by the algorithm. Lines 2-4 initialize the tag count for every POS tag to 0. Then, for every keyword in every article of the dataset, the corresponding POS tag of the keyword is determined. In line 8, The count for that POS tag is increased by 1. Once this terminates, Probability Distribution, P(tag) is determined by dividing the tag count by the total number of keywords.

4.1.2 Extraction

Now we shall proceed on to the extraction algorithm proposed by us Given below is the model for the work we have done.

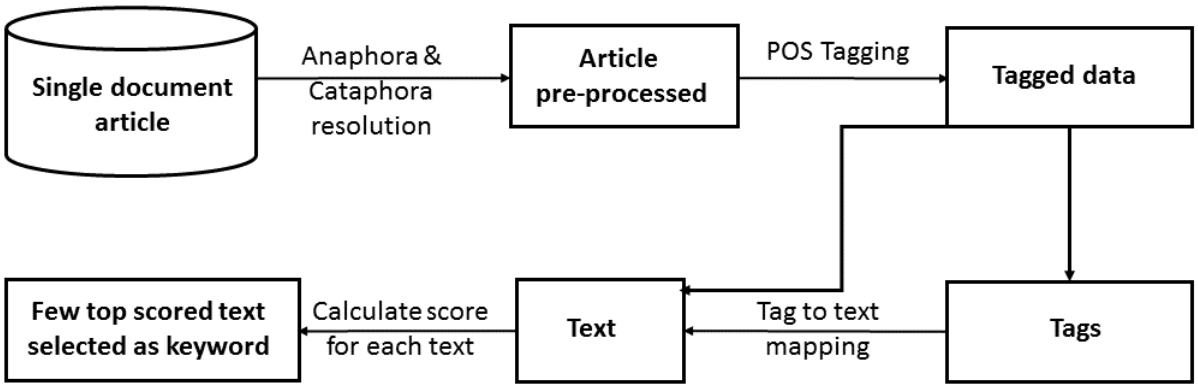


Figure 4.2: Model for Extracting Keywords

We shall deal with the functionalities shown above in the next few sections.

4.1.2.1 Anaphora and Cataphora Resolution

The word Anaphora is derived from Ancient Greek which means The act of carrying back upstream. It involves two parts; Anaphor : The part which is pointing(Reference) and the Antecedent: The part which is being pointed to.

Example:

Sachin isnt out yet but he should be any minute.

In this example, Sachin is the Antecedent and he is the Anaphor. Anaphora resolution deals with matching the anaphor to the antecedent. Hence converting the above sentence into

Sachin isnt out yet but Sachin should be any minute.

Similarly Cataphora, too is derived from Ancient Greek, meaning The act of carrying forward. It involves replacing an anaphor by an antecedent that occurs later in the text and not before as was the case in Anaphora resolution.

Example:

It is tough, It is irritating and it is boring. I hate writing thesis.

Cataphora resolution would resolve the above sentence into:

Writing thesis is tough, writing thesis is irritating and writing thesis is boring. I hate writing thesis.

The file undergoing extraction is first supplied to the Anaphora and Cataphora. This is basically a cleaning step that gets rid of pronouns and empowers the origin of the pronoun. Once every pronoun is replaced with its antecedent, the file is passed to a POS tagger.

4.1.2.2 Parts-of-speech Tagging

The extraction part also undergoes POS tagging. The document to be analyzed is supplied as input after it undergoes Anaphora and Cataphora resolution. After Processing, every word in the article is tagged with its respective part of speech. This document is forwarded to the next section of the extraction algorithm .

4.1.2.3 Keyword Extraction

The output file from the POS tagger is now forwarded to the Model for extraction purposes. Unlike tf-idf, where we keep count of the number of times a particular word has appeared, we keep count of the word-tag pair. i.e [Can, Noun] and [Can, Verb] are treated differently. When a count of the entire document is taken, the keywords are ranked on the basis of the following equation:

$$\text{Score}=P(\text{tag})*\text{Count}(\text{word,tag})$$

where P(tag) is the probability of a tag being a keyword and Count(word,tag) is the number of times the word has appeared in the current document. P(tag) is the same trained probability measure we explained in the previous section.

Algorithm 2: Algorithm to extract Keywords

```
   Data: doc := Input Article
         P(Tag) := Set of Trained Probabilities
         Num_Keywords := Required Number of Keywords
   Result: Keywords[]
 1 res_doc:=resolve(doc);
 2 pos_doc:=pos_tagger(res_doc);
 3 top:=0;
 4 while word in pos_doc do
 5 |   flag:=0;
 6 |   for i ← 0 to top do
 7 |   |   if word.text=wordset[i].text and word.tag=wordset[i].tag then
 8 |   |   |   wordset[i].count:=wordset[i].count+1;
 9 |   |   |   flag:=1;
10 |   |   end
11 |   end
12 |   if flag=0 then
13 |   |   wordset[top + 1].word:=word.word;
14 |   |   wordset[top + 1].tag:=word.tag;
15 |   |   wordset[top + 1].count:=1;
16 |   |   wordset[top + 1].score:=0;
17 |   |   top:=top+1;
18 |   end
19 end
20 for i ← 0 to size do
21 |   wordset[i].score:=wordset[i].count*P(wordset[i].tag);
22 end
23 sort_desc(wordset.score);
24 for i ← 0 to Num_Keywords do
25 |   Keywords[i]:=wordset[i];
26 end
```

Given above is the algorithm for extracting keywords out of an input article. The inputs to the algorithm are The article whose keywords are to be extracted, the number of keywords to be extracted and a probability distribution trained during the training phase. The output of the algorithm will be saved in an array Keywords[].

Now we enter the body of the algorithm. Wordset[] is an array of structures that keeps record of the words that have already been scanned and how many times that word-tag pair has been scanned. Top is the variable that stores the value of the number of words scanned. Line 3 initializes top to 0. The input file is passed through a anaphora and cataphora resolver to get its tags in Line 1. The output file is subjected to POS tagging in line 2. The algorithm then courses through the file, updating existing records on the way and creating new ones when needed. When the algorithm is done parsing the file, the scores are updated. Once the scores are set,

the array is sorted according to the scores of each word-tag pair. The top 5 scores are then extracted out as keywords.

4.2 Summarization

Since we did proceed this far into this topic, we felt it a grave injustice if we left without suggesting a summarization algorithm to wrap this up. Let us take into consideration what we have in hands till now. We have a set of word tag pair keywords as well as their respective scores which we calculated though the formula P(tag)*Count(word,tag).

Our algorithm basically suggests that one derives as many sentences for a keyword from the article as is proportional to the score it received. It can derive these sentences through any means, be it through clustering means or through crude scoring. The added advantage that this algorithm gives is the simple statement that

"Not all keywords are equal"

And what better way is there than to discriminate between these keywords than at the time that they are selected.

Well take an example to explain how this algorithm works. Let us assume we were writing an article on Pollution. Possible keywords would be pollution, destruction, harmful and disease. Suppose we require 20 sentences and the individual scores of the keywords were as follows:

Pollution	4.3
Destruction	0.8
Harmful	2.4
Disease	2.5

Now we shall extract

$$\lceil \frac{(Score\,of\,the\,keyword * Total\,Number\,of\,sentence\,required)}{(Total\,score\,of\,all\,the\,keywords)} \rceil$$

sentences for each of the keywords to get the desired summary.
Hence we extract

Pollution	9 sentences
Destruction	2 sentences
Harmful	5 sentences
Disease	5 sentences

Just knowing the number of sentences to extract, is not good enough. One also needs to know how to extract those sentences. Our algorithm is basically meant to piggyback upon other algorithms. I shall explain here how our algorithm piggybacks upon the two algorithms that were described in section 2.2 .

4.2.1 Piggybacking on FCFS

Let us continue working with the example described above. The algorithm starts extracting sentences from the start of the document. We already know the set of keywords and how many sentences each keyword can bring into the summary. Now suppose, we finished extracting 5 sentences for the Keyword Harmful. The quota for that particular word is over. Hence the word is removed from the keyword list. The algorithm continues until all the keywords are removed the keyword list.

4.2.2 Piggybacking on Text Rank - I

As explained before in section 2.2, the algorithm sorts the sentences in descending order according to the number of keywords each sentence contains. The algorithm now starts popping sentences from the top of this sorted stack. In our algorithm, instead of popping a predetermined number of sentences from the top, if a keyword is present in the popped sentence, the number of allocated sentence is reduced by 1. However if a sentence has no keywords whose allocated sentence is greater than 0, the sentence is rejected. The algorithm continues until the number of allocated sentence for each keyword becomes 0.

4.2.3 Piggybacking on Text Rank - II

This algorithm contains a slight twist to it. Instead of scoring sentences by the number of keywords it contains, we score them using the scores of the individual keywords present in them.
For example :

Pollution is so harmful that many international agencies have come together to battle pollution.
The above sentence has a score of 2*score (Pollution) + score (Harmful).The algorithm then sorts the sentences according to these scores in descending order and extracts the required sentences from the top.

Similarly, our algorithm can be piggybacked over numerous existing algorithm, which gives it a flexibility of use.

Chapter 5

Datasets Used

This project has required the assistance of multiple datasets for various phases of training. This chapter describes in detail the datasets used for the implementation.

5.1 POS Tagging

POS Tagging is a supervised process of tagging each word with its respective grammatical Part-Of-Speech. For our training purposes, we have used a data set that consists of sentences and words. Each word has two tags associated with it. One indicates which tag is associated with it and the other indicated which phrase it belongs to. We are only interested in the first as far as this project is concerned. The dataset is in the form of a text file. The features of an entity (word, tag, and phrase) are separated by spaces and entities themselves are separated by the newline character. The dataset follows the Brown corpus tagging schema that uses a set of the 45 tags given below.

Tag	POS
CC	Coordinating Conjuction
CD	Cardinal Number
DT	Determiner
EX	Existential There
FW	Foreign Word
IN	Preposition
JJ	Adjective
JJR	Adjective, Comparative
JJS	Adjective, Superlative
LS	List Item Marker
MD	Modal
NN	Noun
NNS	Noun, Plural
NNP	Proper Noun
NNPS	Proper Noun, Plural
PDT	Predeterminer
POS	Possessive Ending
PRP	Personal Pronoun
PRP$	Possessive Pronoun
RB	Adverb
RBR	Adverb Comparative
RBS	Adverb Superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb
VBD	Verb, past tense
VBG	Verb, Gerund/Present Participle
VBN	Verb Past Participle
VBP	Verb, Non 3rd Person, Singular, Present
VBZ	Verb, 3rd Person, Singular, Present
WDT	Wh-Determiner
WP	Wh-Pronoun
WP$	Possessive Wh-Pronoun
WRB	Wh-Adverb
#	Pound Sign
$	Dollar Sign
.	Sentence Final Punctuation
,	Comma
:	Colon, Semi Colon
(	Left Bracket
)	Right Bracket
”	Double quote
’	Single Quote

Below we present a part of the file, showing a few entities, describing words and their parts of speech.

Text	Tag	Phrase
Confidence	NN	B-NP
in	IN	B-PP
the	DT	B-NP
pound	NN	I-NP
is	VBZ	B-VP
widely	RB	I-VP
expected	VBN	I-VP
to	TO	I-VP
take	VB	I-VP
another	DT	B-NP
sharp	JJ	I-NP
dive	NN	I-NP
.	.	O

For testing purposes we have a similar file, consisting of words and sentences.

5.2 Document Dataset

For deriving a probability distribution for the tags to be a keyword, we needed a dataset. Articles from the website of The Times of India were used as our dataset. Our dataset included 600 articles ranging from the 1st of January 2015 to 28th of February. Our model was developed and tested against articles again from the The Times of India in the month of March.

Chapter 6

Case Study

Now, we proceed to explain the algorithms explained in the previous section using an example. I have taken for our case study a clipping from The Times of India. The article entitled US drone strike in Pakistan kills 5 militants, is given below:

Pakistani intelligence officials say missiles fired from a US drone struck a compound in the North Waziristan tribal region near the Afghan border, killing five militants.

The two officials, who spoke on condition of anonymity because they were not authorized to brief reporters, said three of the militants killed today were Uzbeks and the other two were Pakistanis. They say the compound had been used by the Pakistani Taliban.

US drone strikes have killed several leading militants in rugged areas of Pakistan over the years, but are deeply unpopular in Pakistan because they have also killed civilians. Pakistan has been waging a military offensive in North Waziristan for nearly a year, trying to drive out Taliban and other militants who have long enjoyed safe haven in the lawless region.

The first step in our algorithm was Cataphora and Anaphora resolution. Once the input underwent the resolution, the output produced is given below. Changes that occurred are underlined to provide ease of understanding.

Pakistani intelligence officials say missiles fired from a US drone struck a compound in the North Waziristan tribal region near the Afghan border, killing five militants.

The two officials, who spoke on condition of anonymity because the two officials were not authorized to brief reporters, said three of the militants killed today were Uzbeks and the other two were Pakistanis. The two officials say the compound had been used by the Pakistani Taliban.

US drone strikes have killed several leading militants in rugged areas of Pakistan over the years, but are deeply unpopular in Pakistan because US drone have also

killed civilians. Pakistan has been waging a military offensive in North Waziristan for nearly a year, trying to drive out Taliban and other militants who have long enjoyed safe haven in the lawless region.

Why this was necessary will be evident in the next part of the algorithm where we tag each word with its part-of-speech. Pronouns tend to deprive its antecedent of its power and since they themselves are hardly ever considered keywords, resolving them will give more power to the ones who deserve them. So now we pass this article to a POS Tagger. The results are as follows

Pakistani—NNP intelligence—NN officials—NNS say—VBP missiles—NNS fired—VBN from—IN a—DT US—NNP drone—NN struck—NN a—DT compound—NN in—IN the—DT North—NNP Waziristan—NNP tribal—NN region—NN near—IN the—DT Afghan—NNP border—NN ,—, killing—VBG five—CD militants—NNS .—.

The—DT two—CD officials—NNS ,—, who—WP spoke—NN on—IN condition—NN of—IN anonymity—NN because—IN the—DT two—CD officials—NNS were—VBD not—RB authorized—VBN to—TO brief—JJ reporters—NNS ,—, said—VBD three—CD of—IN the—DT militants—NNS killed—VBD today—NN were—VBD Uzbeks—NNP and—CC the—DT other—JJ two—CD were—VBD Pakistanis—NNP .—. The—DT two—CD officials—NNS say—VBP the—DT compound—NN had—VBD been—VBN used—VBN by—IN the—DT Pakistani—NNP Taliban—NNP .—.

US—NNP drone—NN strikes—VBZ have—VB killed—VBN several—JJ leading—VBG militants—NNS in—IN rugged—JJ areas—NNS of—IN Pakistan—NNP over—IN the—DT years—NNS ,—, but—CC are—VBP deeply—RB unpopular—JJ in—IN Pakistan—NNP because—IN US—NNP drone—NN have—VBP also—RB killed—VBN civilians—NNS .—. Pakistan—NNP has—VBZ been—VBN waging—VBG a—DT military—JJ offensive—NN in—IN North—NNP Waziristan—NNP for—IN nearly—RB a—DT year—NN ,—, trying—VBG to—TO drive—VB out—RP Taliban—NNP and—CC other—JJ militants—NNS who—WP have—VBP long—RB enjoyed—VBN safe—NN haven—RB in—IN the—DT lawless—NN region—NN .—.

Now we start counting the number of times each keyword-tag pair has come together. A part of the result is shown below. We would suggest comparing it to the above tagged article

Word	Tag	Count
Pakistani	NNP	2
Intelligence	NN	1
officials	NNS	4
say	VBP	2
missiles	NNS	1
fired	VBN	1
from	IN	1

Table 6.1: Sample output after determining counts of word-tag pairs.

The next step in our work would be to calculate the scores of each keyword-tag pair. That is done by multiplying the above result by the probability score of the tag. Part of the results attained is shown below. For better understanding please compare these results with table 3.2

Word	Tag	Score
Pakistani	NNP	0.2554
Intelligence	NN	0.363
officials	NNS	0.5764
say	VBP	0
missiles	NNS	0.1441
fired	VBN	0
from	IN	0

Table 6.2: Sample output after determining scores of word-tag pairs.

After this, we proceed to sort the results derived in the previous section. Given below is the top few results after sorting

Word	Tag	Score
drone	NN	1.089
compound	NN	0.726
region	NN	0.726
officials	NNS	0.5764
militants	NNS	0.5764
US	NNP	0.3831
Pakistan	NNP	0.3831

Table 6.3: List of Keywords derived from the article

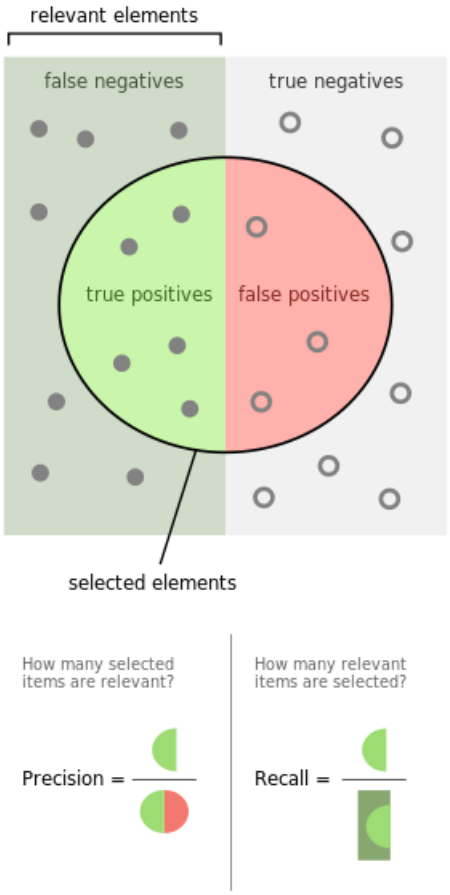
Chapter 7

Evaluation

In this chapter we evaluate the quality of the keywords produced via our suggested algorithm. We shall do so using famous evaluation measures: precision and recall. In section 6.1, we shall define the evaluation measures in detail and in section 6.2, we shall show how well our algorithm performed under those evaluation measures and compare them with existing algorithms.

7.1 Evaluation Measures

In information retrieval scenarios, information retrieved is usually classified under two labels : 'Relevant' and 'Not-Relevant'. One's target is to maximize the number of relevant data and to reduce the number of irrelevant data. To evaluate how good extracted information is, is usually measured by means of two simple measures which have been described in the next subsection.



Credit: *By Walber (Own work), via Wikimedia Commons*

Figure 7.1: Precision and Recall

7.1.1 Precision

Precision is defined as the ration between the number of relevant keywords retrieved by a search and the total number of keywords retrieved by that search. In other words, the precision for a class is the number of true positives (i.e. keywords that have been correctly labelled as such by the algorithm) divided by the total number of elements labeled as belonging to the positive class (i.e. all the identified words of the document regardless of whether they actually were keywords or not). A perfect precision score of 1.0 means that every result retrieved by a search was relevant (but says nothing about whether all relevant documents were retrieved).

7.1.2 Recall

Recall, for our purposes, is defined as the number of relevant keywords returned by our algorithm divided by the total number of actual keywords. In other words, Recall is the number of true positives divided by the total number of elements that actually belong to the positive class. A perfect recall score of 1.0 means that all relevant

documents were retrieved by the search (but says nothing about how many irrelevant documents were also retrieved).

7.2 Evaluation

For testing purposes, we ran our algorithm against a set of newspaper articles again, however this time the headlines wasnt provided to the algorithm. The input was the newspaper clippings and the end target was to extract words that were present in the headlines. On having run the algorithm against the clippings, we got the following results.

True Positives: 178
False Positives: 119
True Negatives: 129

Hence Precision is 0.60 and recall is 0.58
Heres how they compare to existing algorithms

	Tf	KeyGraph	Lexical Chains	Word-Cooccurrence	Bayes	Ours	Tf-Idf
Precision	0.53	0.42	0.45	0.51	0.38	0.61	0.55
Recall	0.48	0.44	0.37	0.62	0.81	0.57	0.61
F1-Measure	0.50	0.43	0.41	0.56	0.52	0.59	0.58

7.3 Discussion

The reason our algorithm works so good is that, it rides on the strong foothold of Tf-idf which is one of the highest scoring performers. Adding to that, the powerful features of Natural Language Processing made it perform better. The concept of Anaphora and Cataphora Resolution too empowered the algorithm to perform better than the existing algorithms. Thus, we managed to take the most powerful features out of statistical and pattern recognition approaches to create a powerful hybrid algorithm. The demerits of this algorithm is that it is much more time taking than some of the existing algorithms.

The proposal on the text summarization algorithm introduces a new concept that encourages people to treat each word with a difference. Some keywords are more relevant to the toppic and some are less relevant to it. That fact has been exploited in the proposed algorithm and what makes it interesting is its flexibility to be piggybacked over most existing summarization algorithms.

Chapter 8

Conclusions and Future Work

8.1 Conclusion

This work has proposed interdependent algorithms in Keyword Detection and Text Summarization. The keyword detection algorithm worked very efficiently in recognizing keywords and had an impressive precision and recall rates. On the basis of the keyword extraction algorithm proposed, a summarization algorithm was proposed that introduced a concept that said not all words are equal, and yet had the flexibility to piggyback over other existing summarization algorithms.

8.2 Future Work

Work can be proceeded in improving the scoring mechanism. The one currently in use is a bit crude and researchers can find that it can be improved drastically. Work can also be done in adding features like sarcasm detection. Sarcasm works best within a context. The sentence would lose its context if it were extracted out into the summary and then the summary would lose any meaning it originally contained.

Appendix A

Software used

We implemented Anaphora and Cataphora Resolution using the Stanford Deterministic Coreference Resolution System developed by the The Stanford Natural Language Processing Group.

This system implements the multi-pass sieve coreference resolution (or anaphora resolution) system described in Lee et al. (CoNLL Shared Task 2011) and Raghunathan et al. (EMNLP 2010). The score is higher than that in EMNLP 2010 paper because of additional sieves and better rules (see Lee et al. 2011 for details).

Rest of the algorithms (existing and proposed) were implemented using C++(gcc-4.3.2). The input and output of the algorithms are stored in txt format and has required extensive use of wordpad.

Bibliography

[1] Inderjeet Mani and Mark T Maybury. *Advances in automatic text summarization*, volume 293. MIT Press, 1999.

[2] Hans Peter. Luhns. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[3] Hui Xu Jie Tang Zhang, Kuo and Juanzi Li. Keyword extraction using support vector machine. *In Advances in Web-Age Information Management, Springer Berlin Heidelberg*, pages 85–96, 2006.

[4] Gonenc Ercan and Ilyas Cicekli. Using lexical chains for keyword extraction. *Information Processing & Management*, 43(6):1705–1714, 2007.

[5] Yutaka Matsuo and Mitsuru Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(01):157–169, 2004.

[6] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223. Association for Computational Linguistics, 2003.

[7] Lee-Feng Chien. Pat-tree-based keyword extraction for chinese information retrieval. In *ACM SIGIR Forum*, volume 31, pages 50–58. ACM, 1997.

[8] Marina Litvak and Mark Last. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24. Association for Computational Linguistics, 2008.

[9] Kuo Zhang, Hui Xu, Jie Tang, and Juanzi Li. Keyword extraction using support vector machine. In *Advances in Web-Age Information Management*, pages 85–96. Springer, 2006.

[10] Gerard Salton, Amit Singhal, Mandar Mitra, and Chris Buckley. Automatic text structuring and summarization. *Information Processing & Management*, 33(2):193–207, 1997.

[11] Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. *Advances in automatic text summarization*, pages 111–121, 1999.

[12] Günes Erkan and Dragomir R Radev. Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479, 2004.

[13] Eduard Hovy and Chin-Yew Lin. Automated text summarization and the summarist system. In *Proceedings of a workshop on held at Baltimore, Maryland: October 13-15, 1998*, pages 197–214. Association for Computational Linguistics, 1998.

[14] John M Conroy and Dianne P O'leary. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407. ACM, 2001.

[15] Rada Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 20. Association for Computational Linguistics, 2004.

[16] Ruslan Mitkov et al. *Anaphora resolution*. Routledge, 2014.

[17] Shalom Lappin and Herbert J Leass. An algorithm for pronominal anaphora resolution. *Computational linguistics*, 20(4):535–561, 1994.

[18] Elaine Rich and Susann LuperFoy. An architecture for anaphora resolution. In *Proceedings of the second conference on Applied natural language processing*, pages 18–24. Association for Computational Linguistics, 1988.

[19] Michel Denber. Automatic resolution of anaphora in english. *Eastman Kodak Co*, 1998.

[20] Ruslan Mitkov. *Anaphora resolution: the state of the art*. Citeseer, 1999.

[21] Ruslan Mitkov. Robust pronoun resolution with limited knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 2*, pages 869–875. Association for Computational Linguistics, 1998.